MD5

MD5

	1000	multiple of 512
MS	**Padding**	**64**

P. T. = 512 -64 = <u>448</u>

MS < 448 → Append 64

PT→ 1024-64= 960 → Append 960

MD5 Cont…

F(B, C, D) - (B ∧ C) ∨ (¬B ∧ D)

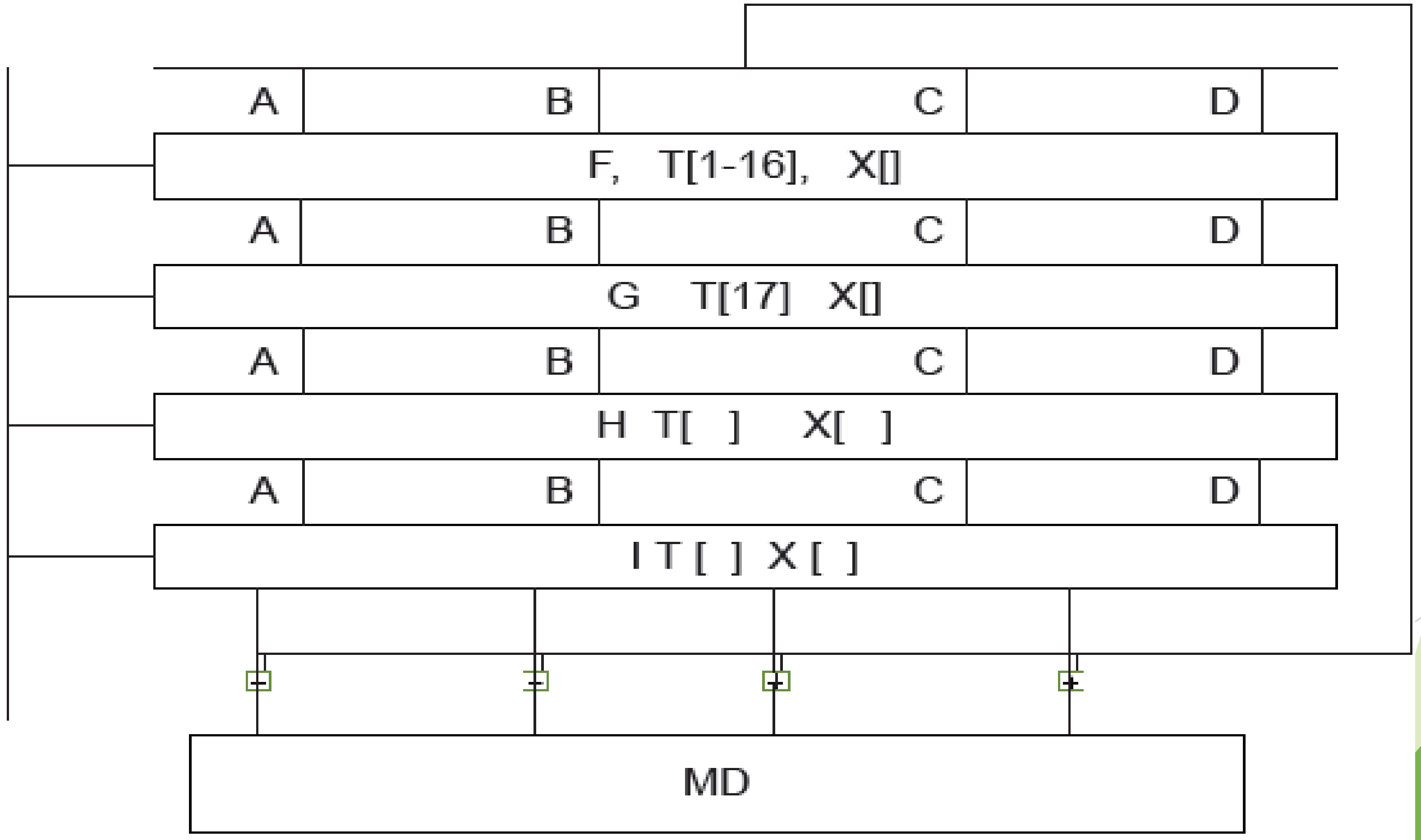
G(B, C, D) – (B ∧ D) ∨ (C ∧ ¬D)

H(B, C, D) – B ⊕ C ⊕ D

I(B, C, D) – C ⊕ (B ∨ ¬D)

$A \leftarrow B \boxed{+} ((A \boxed{+} F_u (B, C, D) \boxed{+} X[\ ] \boxed{+} T[i]) \underbrace{<<}_{\text{Circular left shift}} LS)$

MD5 Cont…



Property 1:

- ***Collision-resistance:*** The first property that we need from a cryptographic hash function is that it's collision-resistant. A collision occurs when two distinct inputs produce the same output. A hash function $H(.)$ is collision-resistant if nobody can find a collision.

- **Collision-resistance:** A hash function H is said to be collision resistant if it is infeasible to find two values, x and y , such that $x \neq y$, yet $H(x) = H(y)$.

Property 2:

- ***Hiding*** The second property that we want from our hash functions is that it's *hiding* . The hiding property asserts that if we're given the output of the hash function $y = H(x)$, there's no feasible way to figure out what the input, x , was.

- ***Hiding.*** A hash function H is hiding if: when a secret value r is chosen from a probability distribution that has *high min-entropy* , then given $H(r \| x)$ it is infeasible to find x .
- In information-theory, *min-entropy* is a measure of how predictable an outcome is, and high min-entropy captures the intuitive idea that the distribution (i.e., random variable) is very spread out.

Commitment scheme. A commitment scheme consists of two algorithms:

- **com := commit(** ***msg, nonce*** **)** The commit function takes a message and secret random value, called a nonce, as input and returns a commitment.
- **verify**(*com, msg, nonce*) The verify function takes a commitment, nonce, and message as input. It returns true if *com* == commit(*msg* , *nonce*) and false otherwise.

We require that the following two security properties hold:

- ***Hiding*** **:** Given *com* , it is infeasible to find *msg*
- ***Binding*** **:** It is infeasible to find two pairs *(msg, nonce)* and *(msg', nonce')* such that *msg* ≠ *msg'* and commit(*msg, nonce*) == commit(*msg', nonce'*)

- Take another look at the two properties that we require of our commitment schemes. If we substitute the instantiation of *commit* and *verify* as well as *H(nonce ‖ msg)* for *com* , then these properties become:
- ***Hiding*** **:** Given H(*nonce* ‖ *msg)* , it is infeasible to find *msg*
- ***Binding*** **:** It is infeasible to find two pairs *(msg, nonce)* and *(msg', nonce')* such that *msg* ≠ *msg'* and H(*nonce* ‖ *msg*) == H(*nonce'* ‖ *msg'*)

Property 3:

Puzzle friendliness. The third security property we're going to need from hash functions is that they are puzzle-friendly. This property is a bit complicated. We will first explain what the technical requirements of this property are and then give an application that illustrates why this property is useful.

Puzzle friendliness. A hash function H is said to be puzzle-friendly if for every possible n-bit output value y , if k is chosen from a distribution with high min-entropy, then it is infeasible to find x such that H(k || x) = y in time significantly less than $2\ n$.

Search puzzle*.* A search puzzle consists of

- a hash function, *H* ,
- a value, *id* (which we call the *puzzle-ID*), chosen from a high min-entropy distribution
- and a target set *Y*

A solution to this puzzle is a value, *x* , such that

$$\mathbf{H}(\, \boldsymbol{id} \,\|\, \boldsymbol{x} \,) \in \boldsymbol{Y}\,.$$